Assignment 3: Fitting Data To Models

Jnaneswara Rao Rompilli - EE20B052

February 18, 2022

Abstract

The content of the assignment is:

- Analysing data to extract information out of it.
- Fit curves along the given data
- To study the effects of noise on the fitting process.
- To plot different types of graphs.

1 Introduction

The data had been generated by adding noise with normal distribution to bessel function

$$f(x) = 1.05J_2(t) - 0.105t + n(t)$$

The first column is the time values and the next nine rows are the noisy values (with different standard deviations) of a function as shown below

`Values of standard deviation = logspace(-1,-3,9)`

2 Assignment tasks

2.1 Importing Data

Numpys loadtxt function is used to load data from input file in the form of 2D array.

```
try:
    data = loadtxt("fitting.dat")
except IOError:
    print("File Path doesn't exist !")
    exit()
```

2.2 Plotting Data and original function

Matplotlib Library is used for plotting the data

```
for i in range(0, k):
    plot(t, data[:, i + 1], label=f"={round(stdev[i],3)}")
plot(t, y_actual, color="black", label="Actual function", linewidth=3)
```

Here, the error in the data follows normal distribution with increasing standard deviation for each column in logscale

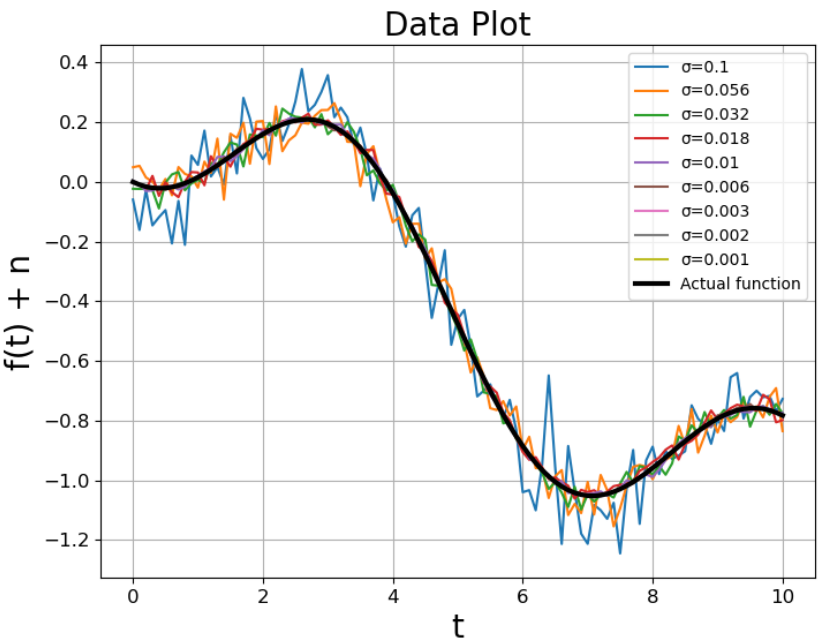

Figure 1: Data plot

2.3 Plotting error bars

```
errorbar(t[::5], data[:, 1][::5], stdev[0], fmt="ro")
plot(t, y_actual, color="black", label="Actual function", linewidth=3)
```

In above code,

```
data[:,1]
```

representsthe 2nd column in the given data.

The graph obtained by plotting every 5th data point with errorbars and the original data is as follows:

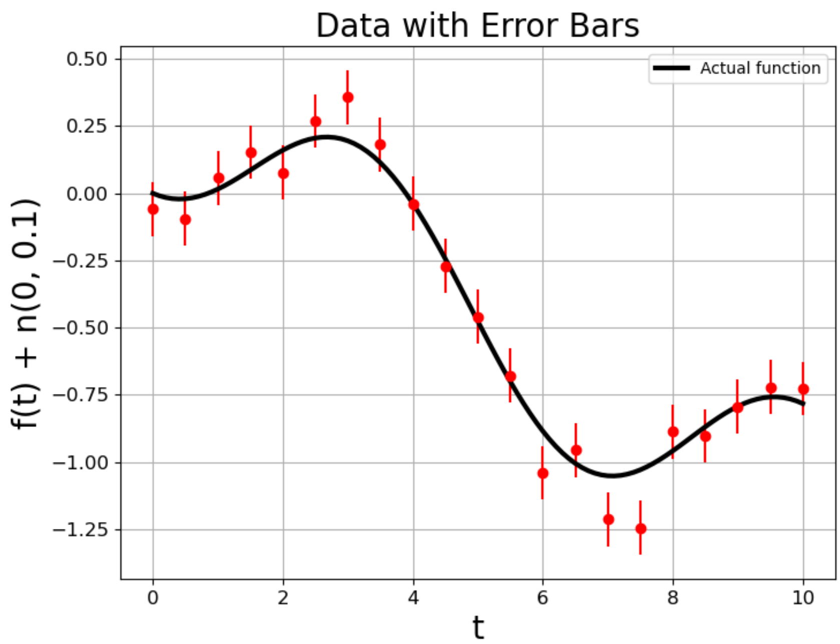

Figure 2: Errorbar plot

2.4 Matrix generation

```
M = np.array((n, 2))
M = c_[sp.jn(2, t), t]
```

2.5 Find RMS error for different A and B values

```
def rms_error(data, t, a, b):
    n = len(t)
    error_value = 0
    for k in range(0, n):
        error_value += ((data[:, 1][k] - g(t[k], a, b)) ** 2) / n
    return error_value

for i in range(0, 21):
for j in range(0, 21):
    E[i][j] += rms_error(data, t, A1[i], B1[j])
```

Formula for finding the RMS error for different values of A and B

$$\varepsilon_{ij} = (\frac{1}{101}) \sum_{k=0}^{101} (f_k - g(t_k, A_i, B_j))^2$$

2.6 Contour plots of error values

```
X, Y = np.meshgrid(A1, B1)

ctr = contour(X, Y, E, 20)
clabel(ctr, ctr.levels[0:5], inline=True, fontsize=10)
```

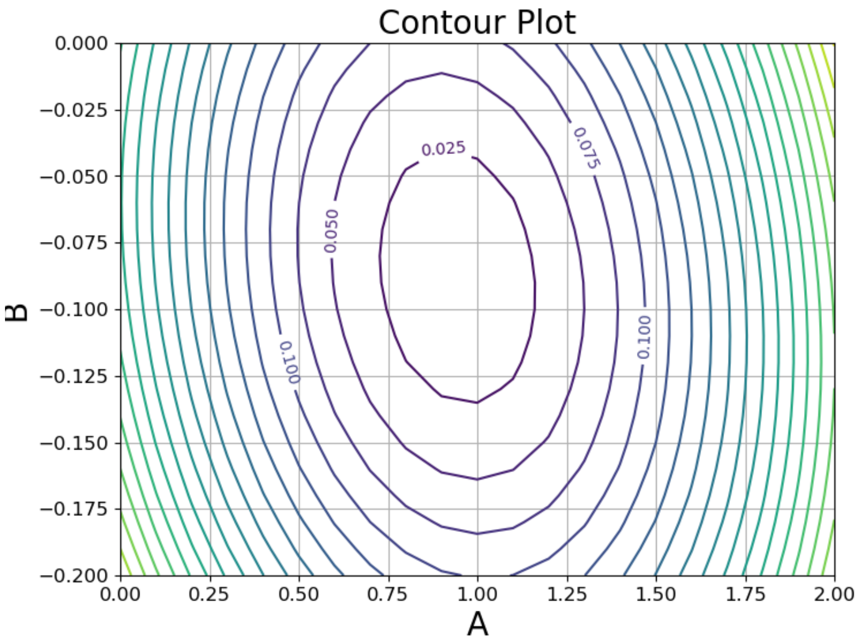

Figure 3: Contour plot

2.7 Estimating A and B coefficients

```
ctr = contour(X, Y, E, 20)
prediction = np.zeros((k, 2))
error = np.zeros((k, 2))
for i in range(1, 10):
    AB = linalg.lstsq(M, data[:, i], rcond=None)
    prediction[i - 1] = AB[0]
    error[i - 1] = abs(prediction[i - 1] - [A, B])
```

We computed the best measure of coefficients A and B by lstsq function in scipy.linalg

2.8 Plotting error in A and B prediction in normal scale

```
plot(stdev, error[:, 0], marker="o", label="A", linestyle="dashed")
plot(stdev, error[:, 1], marker="o", label="B", linestyle="dashed")
```

We calculate the absolute error in A and B values estimated above and plot them. We observe that the error in prediction is genrally increasing with the standard deviation

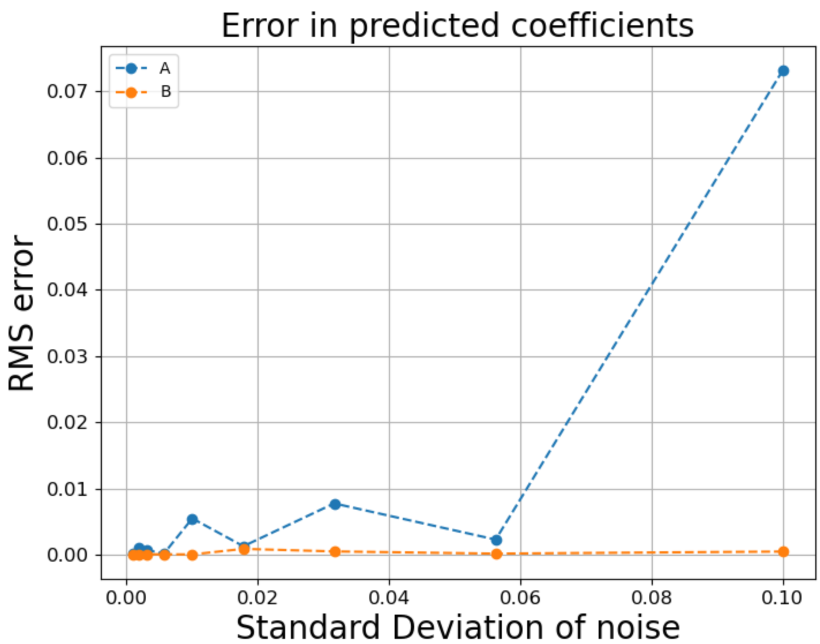

Figure 4: Error of A and B in normal scale

2.9 Plotting error in A and B prediction in Log scale

```
loglog(stdev, error[:, 0], marker="o", label="A", linestyle="none")
errorbar(stdev, error[:, 0], stdev, fmt="ro")
loglog(stdev, error[:, 1], marker="o", label="B", linestyle="none")
errorbar(stdev, error[:, 1], stdev, fmt="go")
```

loglog() function makes the plot with logscale on both x and y axis. We can observe that the graph is not linear

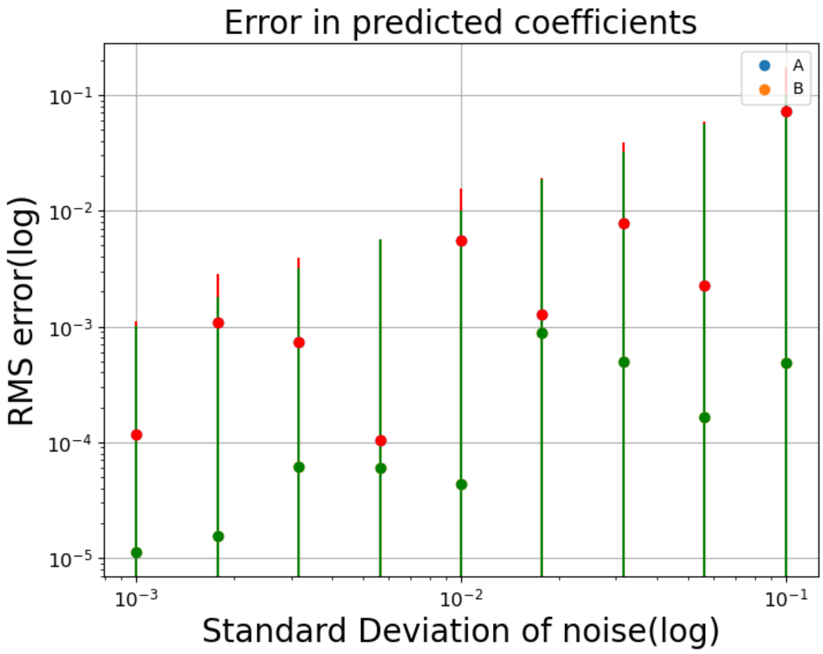

Figure 5: Error of A and B in log scale

3 Conclusion

Given the data with noise added, we calculated the actual coefficients using Least Sqaure Fitting method in Scipy Library and found that error in out prediction is increasing with standard deviation of noise added Also from the above plots we found that graph is not linear in normal scale and also not linear log scale too